**CSCE 629 Lab 5**

**Winter 2019**

**Web App Attacks – Session Cloning / Burp Proxy / Injection / SET**

**Assigned: Lesson 15, 29 Jan**

**Due: Lesson 19, 5 Feb, 1400**

You will work with your partner and submit one solution.

Because you’ll be installing the Burp certificate in the Trusted Root Certification Authorities store, I highly recommend using the Windows 7 VM provided. Also, we’ll be deleting cookies, and I don’t want you to inadvertently delete one of your real cookies on your Windows 10 workstation. Therefore, these instructions are tailored to Internet Explorer (IE) and Windows 7.

**1. Stealing a session - Amazon.com**

**Setup**

You will need to unhide protected operating system files and display hidden files to see the cookies. You will also need to search within file contents.

* **Windows 7**
  + Open Windows Explorer 🡪 Tools 🡪 Folder Options 🡪 View tab 🡪
    - Select “Show hidden files, folders, or drives”
    - Uncheck “Hide protected operating system files”.
  + Open Windows Explorer 🡪 Tools 🡪 Folder Options 🡪 Search tab 🡪
    - Select “Always search file names and contents”
    - Select “Include subfolders in search results when searching in file folders”
    - Select “Find partial matches”
    - Select “Don’t use the index when searching in file folders for system files”
    - Select “Include system directories”
  + Windows 7 cookies are in C:\Users\[username]\AppData\Roaming\Microsoft\Windows\Cookies
    - Type “shell:cookies” in the Windows Explorer address bar to take you there.

Although not required, I suggest setting your IR default home page to www.amazon.com.

***First you’ll delete old cookies and create an account on Amazon which will store new cookies on your computer.***

Both students (victims):

* Close IE.
* Search for Amazon cookies in your cookie directory. Navigate to your cookies folder and search for files containing “amazon” (it will be in the file content—not the filename). Delete all cookies found. This site (<http://www.wikihow.com/View-Cookies>) may help if you are not using IE.
* Open IE.
* Go to Amazon ([www.amazon.com](http://www.amazon.com)) and create an account. The following screenshots (Figures 1-3) should help. The screenshots may differ slightly.

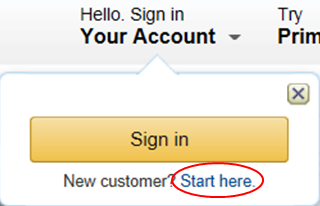


Figure 1. Create an account on Amazon

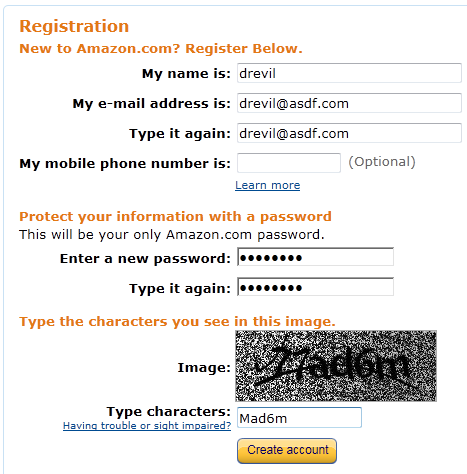


Figure 2. Setting up your credentials on Amazon



Figure 3. Successful account creation

Both students (victims):

* Add a couple of items to the Shopping Cart of this new Amazon account.
* Do not log out of Amazon.
* Close IE.
* Copy the Amazon cookie file that was just created to your team folder.

Both students acting as Blackhats:

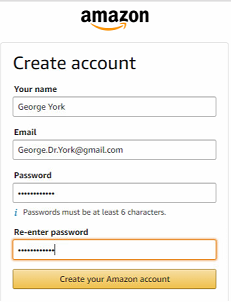
* “Steal” (i.e., copy) your partner’s cookie file and paste it to any directory on your computer.
* Open the cookie file in WordPad/Notepad++. It should look similar to Figure 4.
* Notice there are several fields in the file as indicated by the red arrows. Your fields may be in a different order. Also notice there are values between the name of each field (e.g., session-id-time) and the string “amazon.com/”. These values are sent back to Amazon inside an HTTP cookie header as shown in Figure 5.

Figure 4. Amazon cookie file

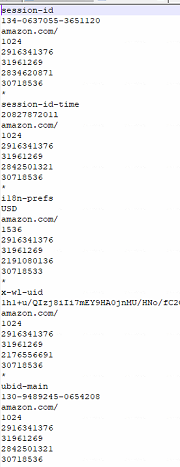
**Cookie: session-id=133-3070606-6074263; session-id-time=2082787201l; ubid-main=134-1524539-0274701**

Figure 5. Amazon Cookie. Notice there are no carriage returns; this is one long line.

**Note:** both students created their own Amazon accounts per the instructions on the previous pages. The cookie files were copied to each computer to have access to the needed fields for the following questions.

Below is a screenshot of the creation of George York’s Amazon account:

Below is a screenshot of RedLeader’s cookie file on the left, and Dr. George York’s on the right:

**a. Speculate on how we know this is the cookie header format used by Amazon? In other words, how can we discover Amazon uses the cookie format shown?**

We can discover that Amazon uses the cookie format shown by sniffing traffic using a tool like Wireshark and inspecting relevant packets. All of the fields used would be indicated in the Set Cookie header in the packet from Amazon to the client on the initial connection between client and Amazon.com

* Create a cookie header in WordPad/Notepad++ using the example in Figure 5 as a template by replacing the highlighted values with your partner’s cookie values. There are no carriage returns in the template.

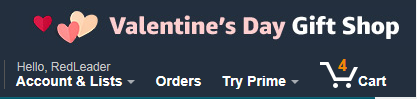
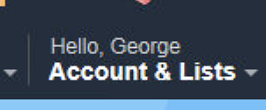
Below is the spoofed cookie header for Dr. York:

Below is the spoofed cookie header for RedLeader:

***Now start Burp Proxy and configure IE to use the proxy for all traffic.***

* Download (<https://portswigger.net/burp/communitydownload>) and start Burp Proxy
  + Start the proxy listener on 127.0.0.1:8081 (Yes, use port 8081 because WebGoat uses 8080)
    - Proxy tab 🡪 Options tab 🡪 Proxy Listeners section 🡪 click on address 🡪 Edit 🡪 make changes 🡪 OK 🡪 Click the Running box next to the interface
  + Turn intercept off.
* Open IE and configure a proxy connection
  + Settings icon 🡪 Internet Options 🡪 Connections 🡪 LAN Settings
  + Check the box next to **Use a proxy server for your LAN** and enter 127.0.0.1 (assuming you’re running the proxy on the same machine as IE) as the address and port number 8081
* You need to install Burp Suite’s CA certificate. If you do not, IE complains about the Burp certificate not being trusted and blocks your connection.
  + Follow the instructions at https://support.portswigger.net/customer/portal/articles/1783075-installing-burp-s-ca-certificate-in-your-browser.
* Test the proxy connection by surfing to www.amazon.com. Since you are not intercepting anything yet, the proxy should be completely transparent and the Amazon page should be displayed.

**b. Are you greeted by name or just “Hello. Sign in Your Account”? Why?**

* After testing the proxy connection using the given Burp settings, we navigated to Amazon.com. We were greeted by name because our browser sent the stored cookie file to Amazon. This allowed the site to connect our previous account/browsing with our current session.
* Below was RedLeader’s greeting from Amazon:
* Below is George York’s greeting:

***Take a look at what Burp has to offer.***

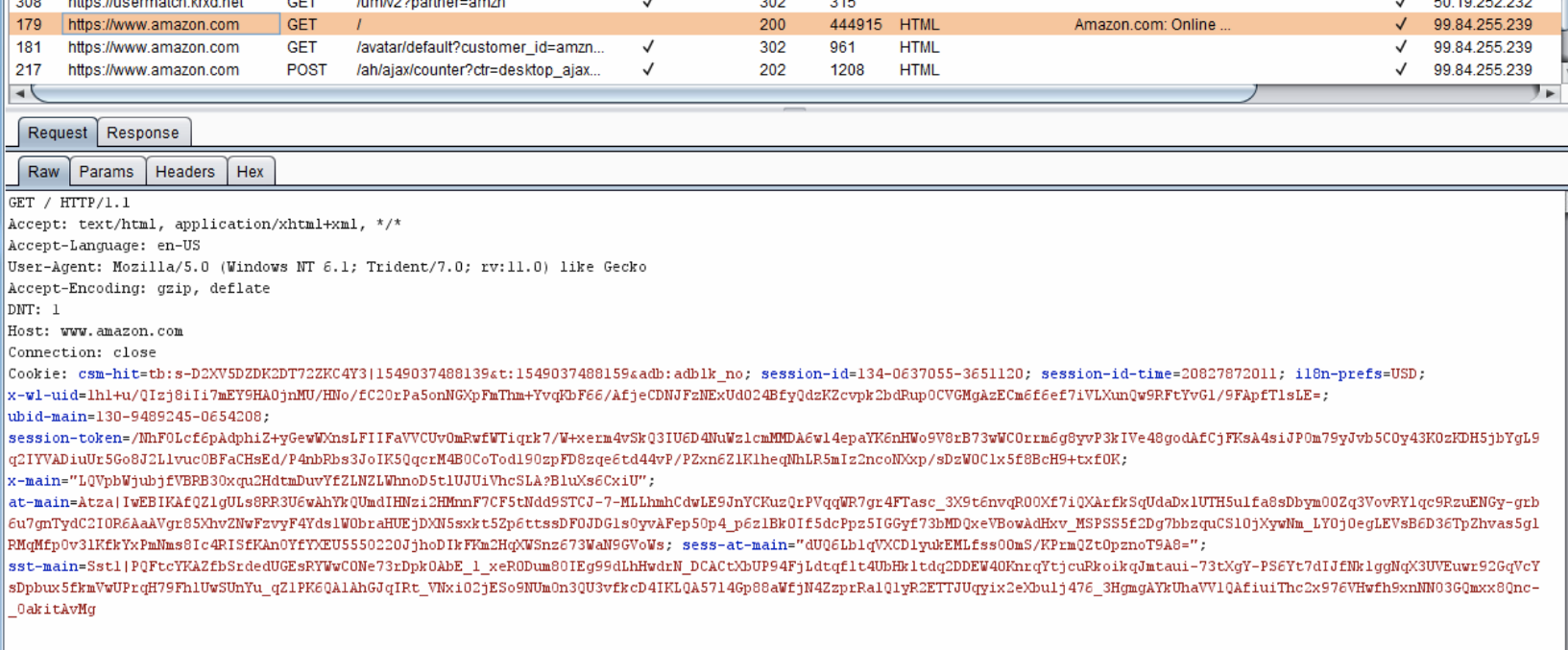
* Within Burp proxy, click on the HTTP history tab and take a look at the HTTP requests to and responses from the Amazon server. It is capturing all traffic over the connection. **[Screenshots not required]**
* Select the first packet sent to the Amazon server. This should be the GET request.
  + Ensure the “Request” and “Raw” tabs are selected in the lower section of the window.

**c. Do you see a Cookie header in the GET packet? Why or why not?**

* Also click on the “Params”, “Headers” and “Hex” tabs to see different views Burp provides of the data within the HTTP request.
* Ensure the GET packet is still selected and click the Response and Raw tabs. This is the response to the GET.

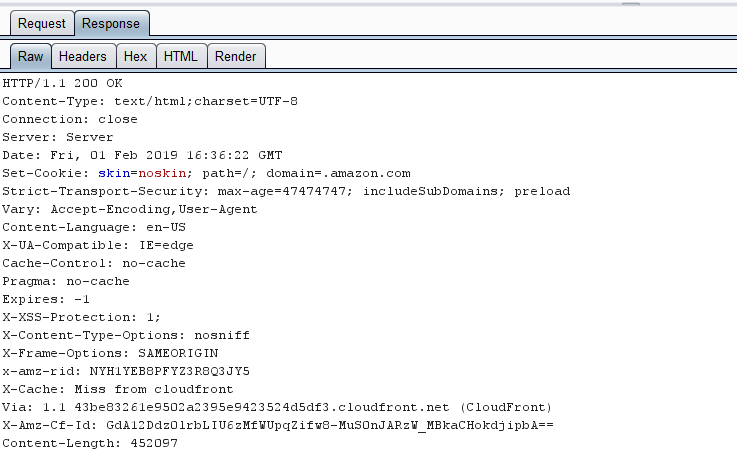
We see a Cookie header in the GET packet. We have this because there is an Amazon cookie stored on our device. When we send a request to Amazon.com, our browser adds the stored cookie information. The header is shown below, outlined in red for George York.

**Note –** RedLeader had a Cookie header in the GET packet, but is not shown for redundancy.



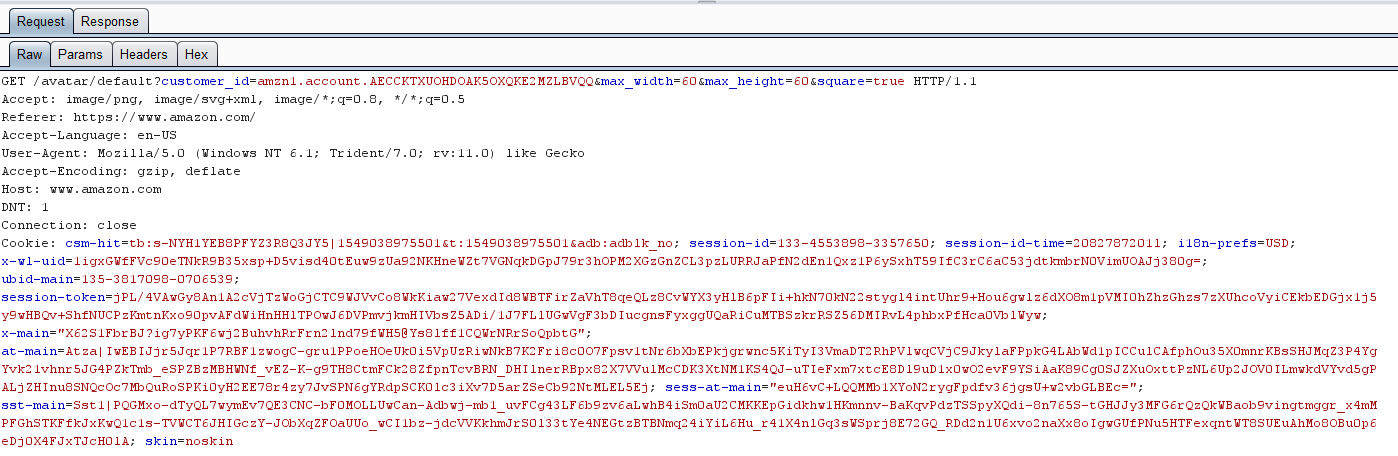
**d. Do you see a Set-cookie: header? Why or why not?**

Yes, we see a set-cookie header in our GET Response from Amazon. This Set-cookie header updated information in our cookie file, specifically setting skin= noskin:



* Click on the other tabs (Headers, Hex, HTML, Render) for the response to the first HTTP packet.
  + Viewing the “HTML” and “Render” tabs can take some time so be patient. Also, the “Render” option doesn’t always yield an exact replica of what you see in IE.
* Select another (second or third) request sent to Amazon and click on the “Raw” and “Params” tab to see the cookie values your browser is sending to Amazon.

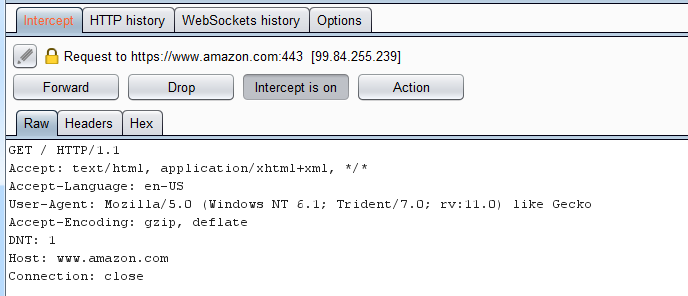
Below is the second HTTP GET request from RedLeader to Amazon.com: the cookie values sent are outlined in red.



***Now let’s intercept the GET request to Amazon…***

* Close IE.
* Delete any existing Amazon cookies in your cookies directory.
* Close and then start Burp. Set the proxy port to 8081 and start the listener (check the Running box).
* Ensure that intercept is on and you are intercepting all requests. No need to intercept responses.
  + By intercepting all requests to Amazon, you can change everything your browser sends to the server.
* Open IE and surf to www.amazon.com.
  + Within Burp, the Proxy and Intercept tabs should turn red indicating the tool has intercepted a request. Click on the Intercept tab and Raw tab to inspect the HTTP request being sent to Amazon. Since you deleted the Amazon cookie, you should not see a cookie header in the request.
  + Click Forward to send the request to the server. Notice you will have to click Forward several times to completely load the Amazon page. You could also turn off Intercept once you’ve seen the items of interest.

The below screenshot shows the intercepted request, with no Cookie header because the Cookie file was deleted:



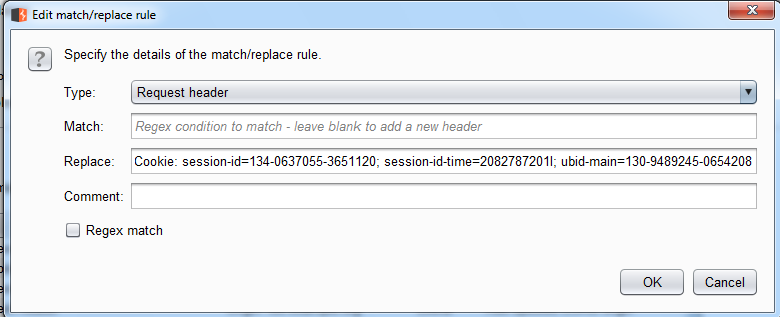
***… and insert the victim’s cookie information (header) so we can masquerade as the victim.***

* Close IE.
* Delete any existing Amazon cookies in your cookies directory.
* Close and then start Burp. Set the proxy port to 8081 and start the listener (check the Running box).
  + Turn intercept off.
  + Click on the Options tab and scroll down to “Match and Replace”
  + Click Add and create the following rule where the “Replace” field is the entire (one long string) modified cookie header you created earlier.

|  |  |  |
| --- | --- | --- |
| type | match | replace |
| request header | <<leave blank>> | **Cookie: x-wl-uid=1Nv7V…** |

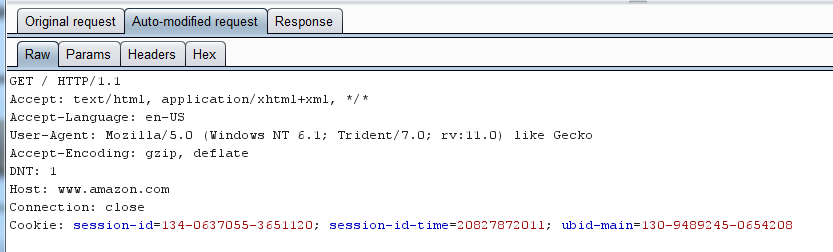
* + According to the Burp documentation, “If a blank matching expression is specified, then the replacement string is added as a new header. This feature is useful to automate certain application attacks, such as manipulation of cookies or URL query string fields.” This essentially adds the Cookie header to all requests sent to amazon.com so you don’t have to intercept all individual requests.
* Burp will add the Cookie header to every HTTP request sent from IE. This includes all requests to all webpages that Amazon might be using (e.g., ad websites).
* Open IE and surf to www.amazon.com.
* In Burp, click on the GET request sent to Amazon. Notice how the tabs changed; the tabs are now Original request, Auto-modified request, and Response. Click on the Original request and Auto-modified request tabs.

We utilized the rule below to spoof George York’s Amazon session from RedLeader’s computer:



**e. Do you see the cookie header inserted by Burp when you view the Auto-modified request?**

* Yes, we see the cookie header inserted in the Auto-modified request, shown below:

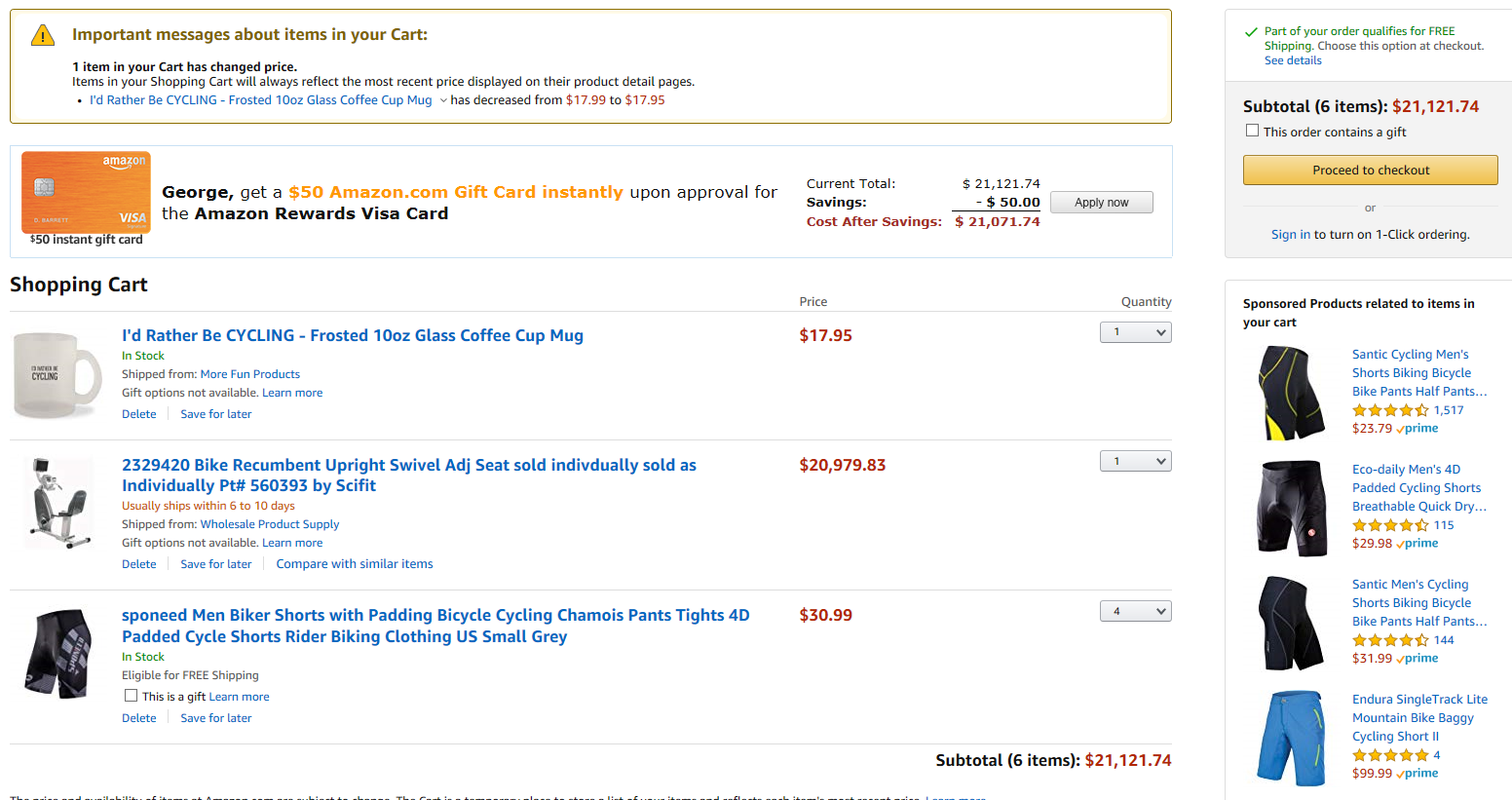


**f. Are you greeted by your partner’s name or just “Hello. Sign in Your Account”? Why?**

* We were both greeting by our partner’s name when using their spoofed cookie information. Their information is displayed because the server only associates a session with the Cookie information provided by the client. For example, the RedLeader client sent George York’s cookie information, convincing the Amazon server that it was George York.

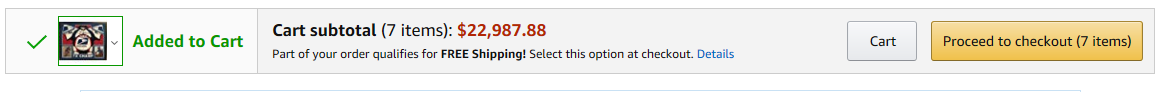
**g. Can you see your partner’s Shopping Cart?**

* Yes, below is the screenshot of George York’s shopping cart when viewed from Red Leader’s computer:



**h. Can you add items to your partner’s Shopping Cart? Is so, provide a screenshot of the additional item. If not, speculate why?**

* Yes, we can add items to the Shopping Cart. The additional item added to George York’s shopping cart is shown below (above there are 6 items due to 4 pairs of cycling shorts):



**BONUS:** For up to 10 bonus points, demonstrate a session-stealing attack against a different website clearly documenting each step.

**2. Moon phases**

Now you will see how to manipulate **data** (not the headers as in problem 1) as it flows across a connection. This problem involves the Blackhat intercepting data sent from the victim and modifying the data.

Blackhat:

* Open Burp.
* Ensure you are listening on all interfaces: Proxy 🡪 Options 🡪 Under Proxy Listeners 🡪 Edit 🡪 Select All interfaces and bind to port 8081. The interface entry should change to \*:8081. This allows other computers to use your computer as a proxy. Start the listener (check the Running box).
* Intercept both requests and responses: Proxy 🡪 Options 🡪 ensure “Intercept Client Requests” and “Intercept Server Responses” are checked.
* Turn intercept on.

Victim:

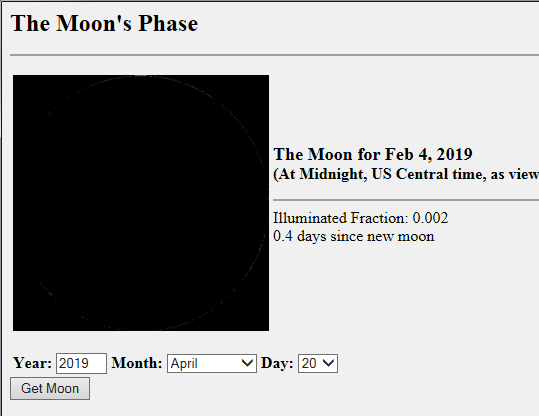
* Open IE.
* Set the proxy address as the IP address of the computer running Burp and the port to 8081.
* Surf to www.briancasey.org/artifacts/astro/moon.cgi.
* Notice the Blackhat will have to click Forward for all requests and responses.
* Once the entire webpage is displayed, select a date in the future and click Get Moon.

Blackhat:

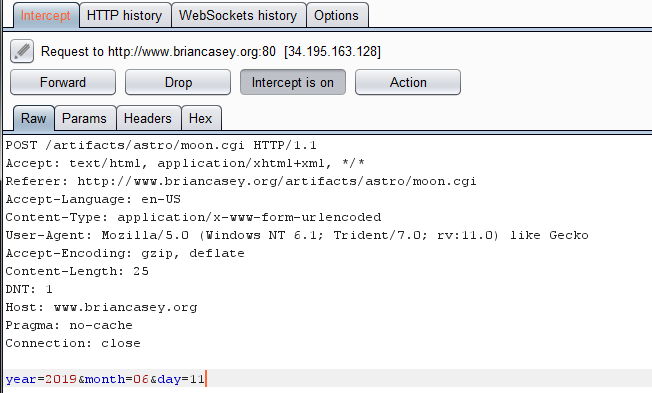
* Change the date **sent to the server** and watch the result on your partner’s computer. For example, assume your partner entered 31 Dec 2024 in the form field on the webpage. Your assignment is to change this date before it is sent (posted) to the server.
  + To expedite the process, you can turn intercept off once you have made your changes to the post message; this forwards the modified packet, and you will not have to continually click Forward. If you do turn off intercept, remember to turn it back on for later steps in this assignment.

1. **Did the returned webpage have the modified date and corresponding image?**

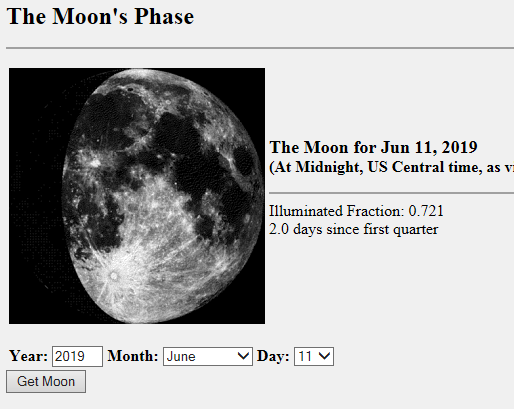
* The returned webpage had the modified date and corresponding image.
* The original request sent is below:



* Using the given settings for Burp Proxy, the Blackhat received the following request, and changed the date from April 20th, 2019 to June 11th, 2019.



* The following web page was what the Victim received:



Victim:

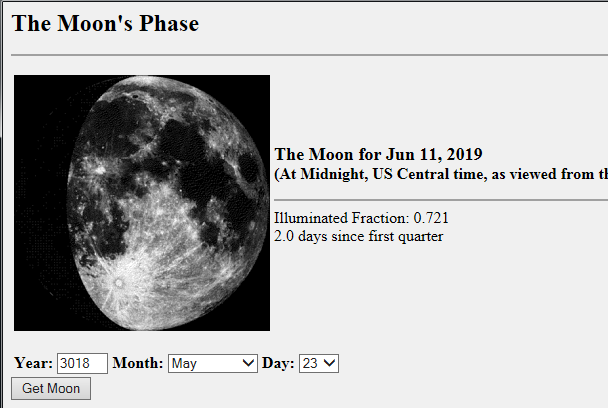
* Once again select another date and click Get Moon.

Blackhat:

* Let the **request** proceed to the server without modification. However, you must **modify the response** returned from the server to change the date displayed to the browser as well as some other text/images displayed. An example is shown in Figure 6.

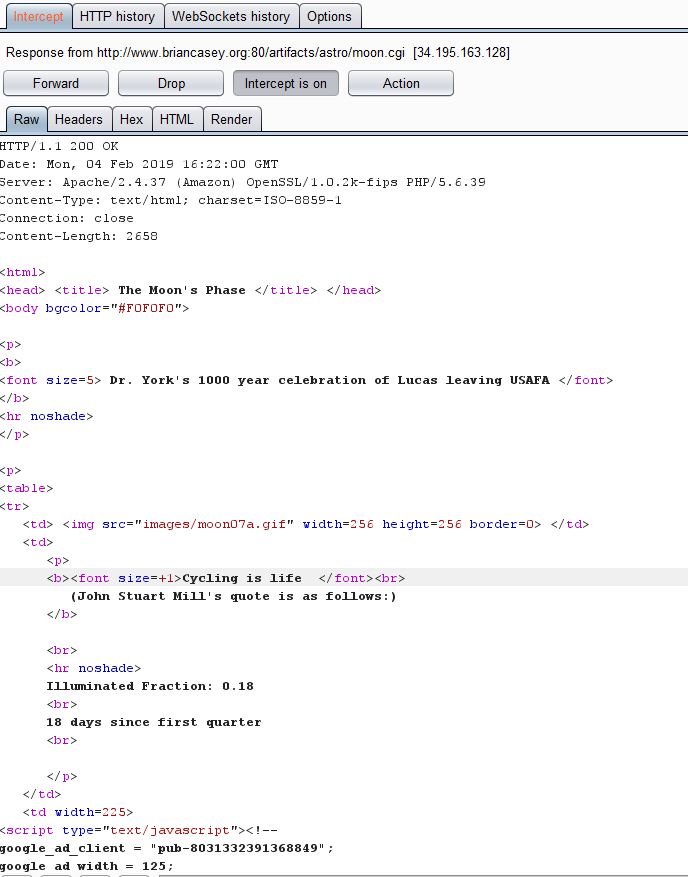
**b. Did the webpage displayed in the browser contain the modifications? Provide a screenshot of the Burp page containing the Blackhat’s modifications and the resulting webpage as seen by the victim.**

* The webpage displayed in the browser contained the modifications.
* The victim sent the following request on the website:

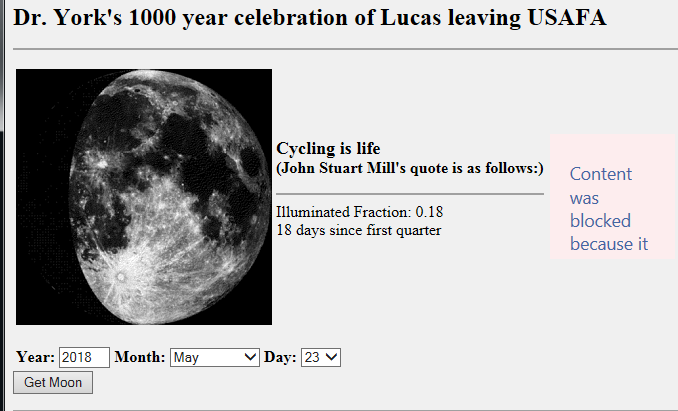


**Burp Settings used, the following settings were used for all of Question 2:**

* Listening on interface \*:8081
* Intercepting requests
* Intercepting responses
* The screenshot below shows the modifications made by the Blackhat in Burp: Not pictured is the requested year was changed from 3018 to 2018.



* Result shown in the victim’s web browser:



**3. WebGoat -- Injection Attacks**

For this assignment, each student will run WebGoat, a web security teaching application from the OWASP. WebGoat consists of a variety of lessons and provides hints to help when you’re stuck. Although I ask you to minimize your use of hints, it’s fine to use them, but some have been known to be incorrect. You may have to use a web proxy for some of the exercises.

* I recommend installing WebGoat on the same computer that has Burp (e.g., Win7-32-629).
* Follow the instruction on the slides to download and install Java and WebGoat.
* You need to execute the jar file using java from command line. I suggest browsing to the jar file in Window Explorer and opening a command shell.
  + [Pro Tip: To open a command shell window in any folder, hold down the shift key and right-click in a blank area in the folder, and select “Open command window here”.]
* Enter **java -jar webgoat-container-7.1-exec.jar**
* Open IE and browse to <http://localhost:8080/WebGoat>. WebGoat uses port 8080 as its server; this is why we kept changing Burp’s port to 8081.
* If Burp is not seeing the traffic add a period after “localhost” as shown <http://localhost.:8080/WebGoat>.
* Sign in as guest.

For each of the WebGoat exercises listed below,

**i. Explain how you solved the lesson, including any and all input (including form fields and headers) you used to solve the exercise (usernames, passwords, form input, URLs, etc.). Also include any proxy settings you may have used.**

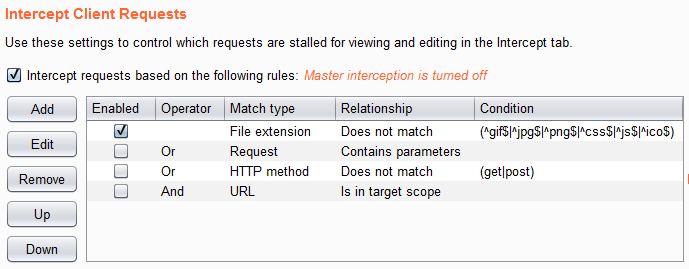
**ii. Provide screenshot of your process as well as the “success” screens.**

**iii. List the Hints you used for each exercise.**

For all exercises, you are not required to perform any attacks beyond the first (e.g., attack with defensive mode enabled, parameterized attack).

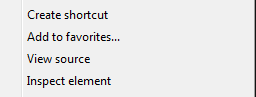
**No hints were used for any of the problems.**

**For all questions that utilized a Proxy, the following settings were used:**

* Listening on loopback interface: 127.0.0.1:8081
* Intercept requests

1. Code Quality 🡪 Discover Clues in the HTML

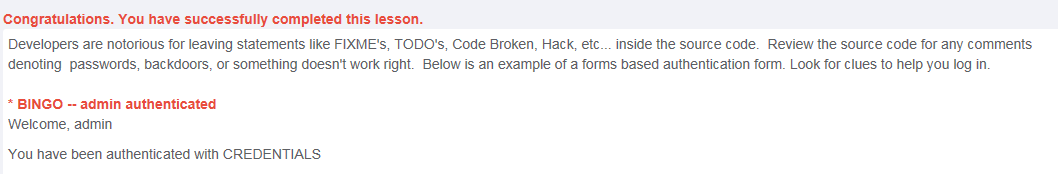
* Inspected the source code of the web page:



* Found a FIXME on the resulting code:



* Signed in using credentials admin:adminpw

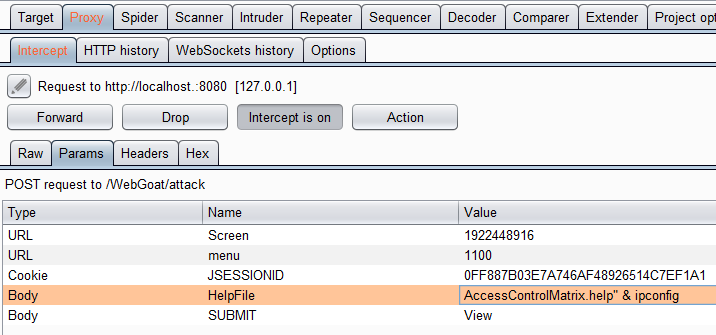


1. Injection Flaws 🡪 Command Injection.

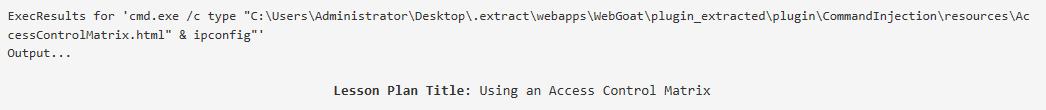
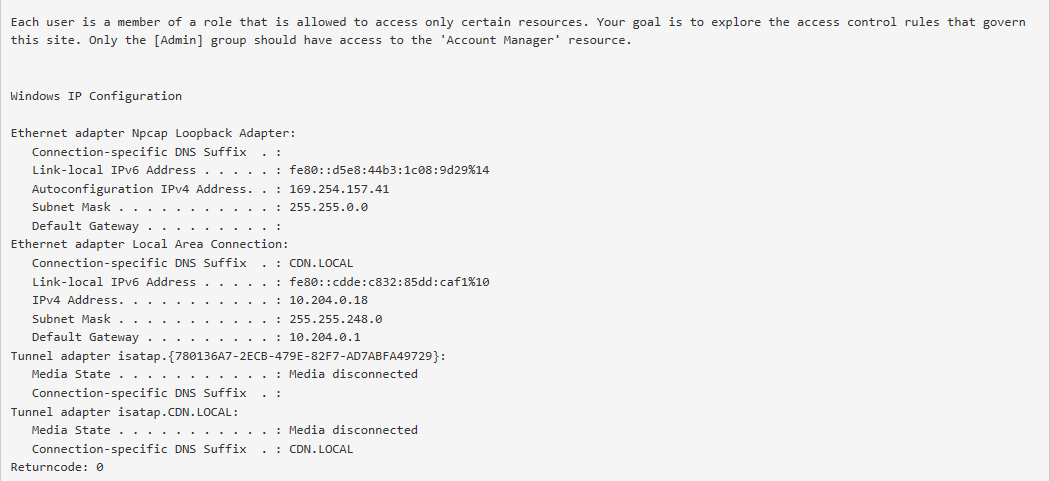
* Configured Burp Proxy using request intercept and the loopback address
* Intercepted a request to view “AccessControlMatrix.help”

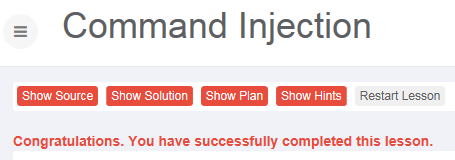


* Modified the request to finish the string and add a command:



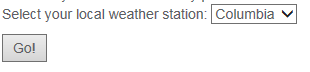
* Output produced:



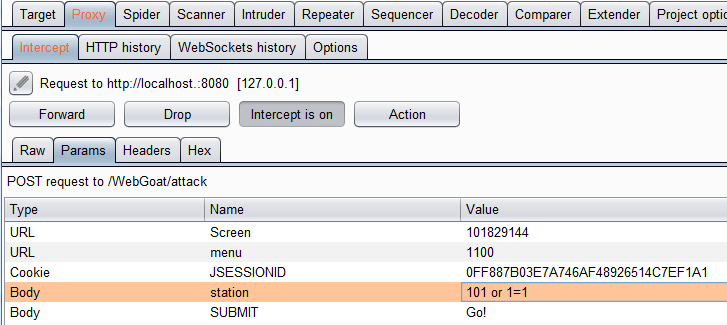
* Success:

1. Injection Flaws 🡪 Numeric SQL Injection.

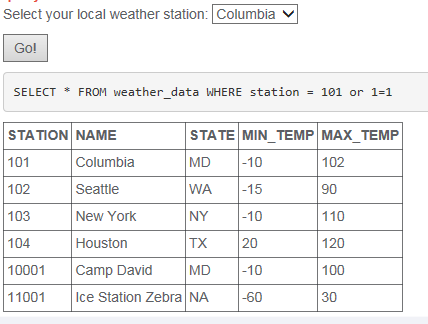
* Submitted a request to the server

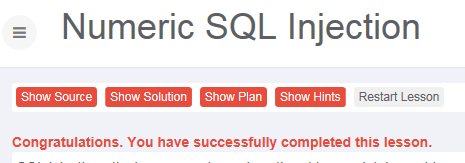


* Used Burp Proxy to intercept/modify the request by appending “or 1=1”:



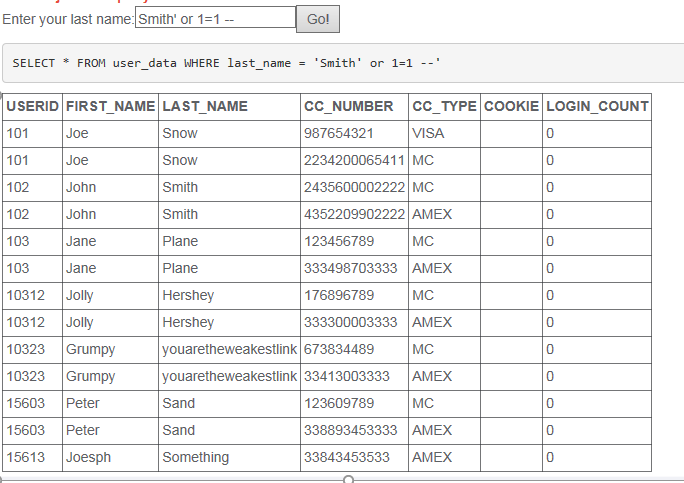
* Output displayed Stations 101-104, 10001, and 11001:



* Success:

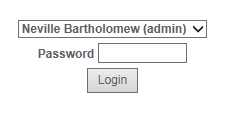
1. Injection Flaws 🡪 String SQL Injection (not “LAB: SQL Injection”).

* Submitted a query using string “Smith’ or 1=1 –“



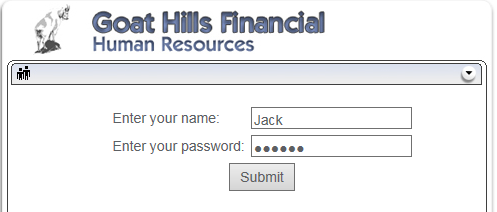
* Success:

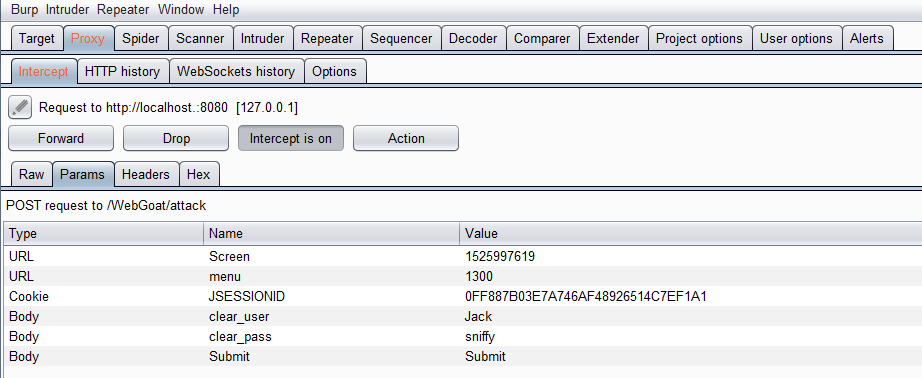
1. Injection Flaws 🡪 LAB: SQL Injection – Stage 1 String SQL Injection only

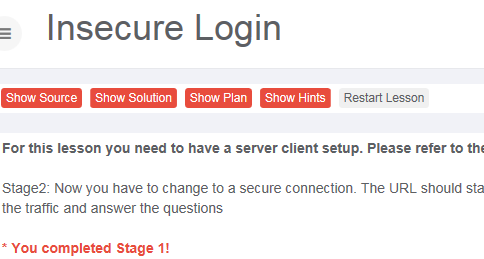
* Opened Burp proxy using request intercept and the loopback IP
* Submitted a login request using Neville and a blank password
* Modified the request to submit a blank password with “or 1=1”, and commenting out the remainder of the line.



1. Insecure Communication 🡪 Insecure Login – Stage 1 only

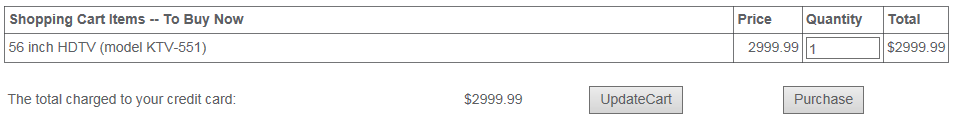
* Setup Burp Proxy to intercept packets using request intercept and the loopback IP
* Submitted the request given on the page:
* Inspected the request packet to find cleartext username and password:



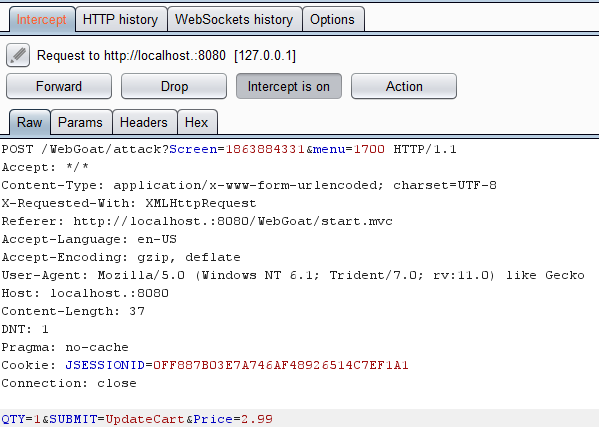
* Submitted sniffy as the password:
* Success:

1. Parameter Tampering 🡪 Exploit Hidden Fields

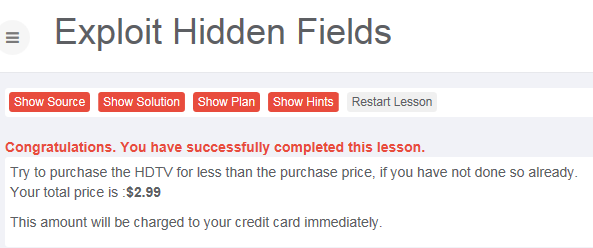
* Turned on Burp Proxy using request intercept and the loopback IP
* Selected UpdateCart:



* Modified the request in Burp Proxy with a new price:



* Success:



**4. Social Engineering Toolkit (SET)**

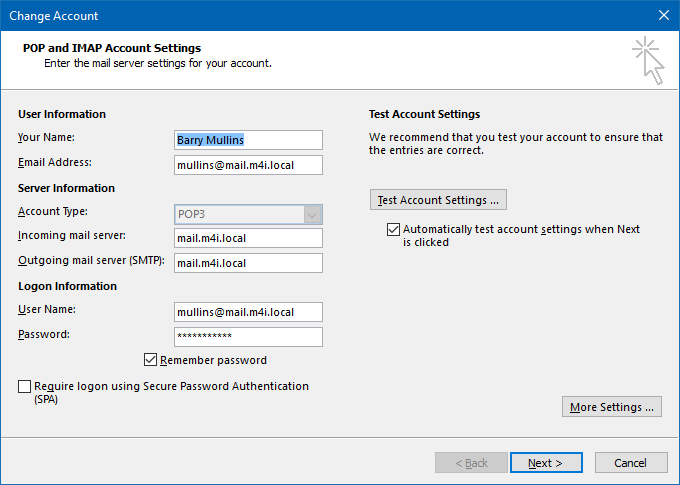
M4I is proud to announce the activation of their mail server (mail.m4i.local); this POP/SMTP mail server is only accessible within the CDN network. Your email address is your last name followed by “@mail.m4i.local” (e.g., mullins@mail.m4i.local).

To configure Outlook:

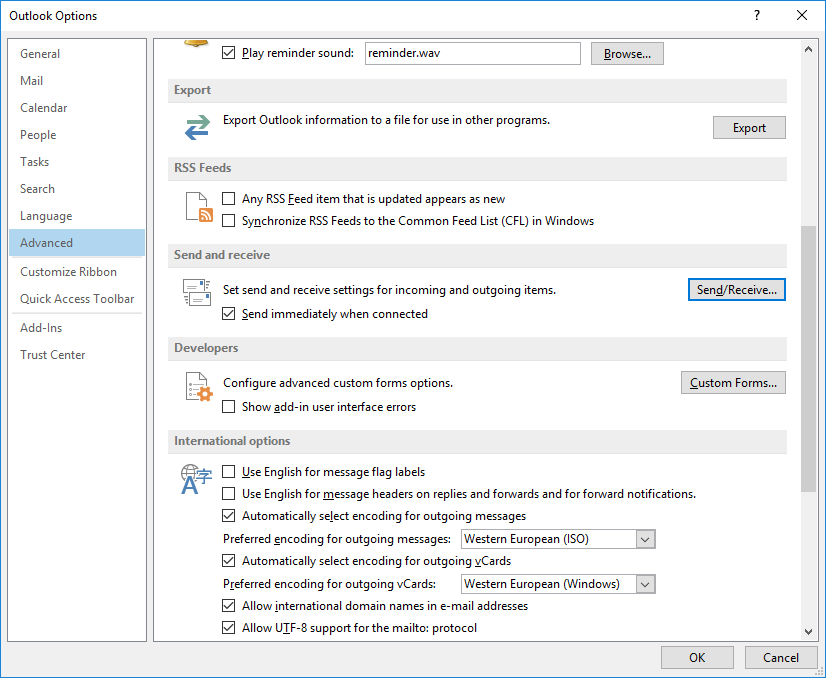
1. Open Outlook.
2. Set up your email account. File – Account Settings – Email tab – New… -- E-mail Account – Manually configure server settings or additional server types – Internet E-mail or POP
   1. Your Name: **First Lastname**
   2. E-mail address: **<<lastname>>@mail.m4i.local**
   3. Account type: **POP3**
   4. Incoming mail server: **mail.m4i.local**
   5. Outgoing mail server: **mail.m4i.local**
   6. User name: **<<lastname>>@mail.m4i.local**
   7. Default password: **Password!123**
   8. More Settings – Outgoing Server tab – Check “My outgoing server (SMTP) requires authentication”
   9. OK
   10. Close
3. Finish

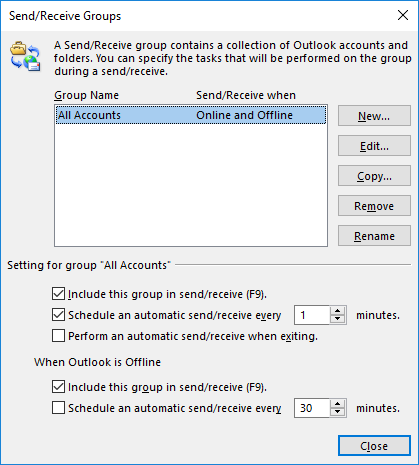
**BONUS:** For up to 15 bonus points, create and execute a second SET attack.

Here is an example:



Now set your email client to send/receive mail every 1 minute so you don’t have to manually check email and you receive my emails/tips sooner. Within Outlook, click File 🡪 Options 🡪 Advanced 🡪 Send/Receive 🡪 Schedule an automatic send/receive every 1 minute.





You cannot keep the default password for your email account. Send your new password to mullins@mail.m4i.local, and I will reset your account. I know this sounds like a phishing attack, but it isn’t. ☺

This task requires you to replicate the SET attack presented in class. Only one person is required to act as the Blackhat for this problem, which gives you a taste of SET.

Blackhat:

* Generate malicious payload and email it to your partner (victim) using convincing language in the email and the payload as an attachment (rename the payload file to match the request in the email).
  + Hint: through reconnaissance you learned the victim uses Outlook for email. Therefore, you may have to compress the .exe file otherwise the victim’s Outlook client will block the attachment. I like 7-zip (http://www.7-zip.org/).
* List all the options / commands used (e.g., selected option 1 for Social-Engineering Attacks). You do not need to provide screenshots if you use the same options shown in the slides.
* Provide a screenshot of the mail message just before it is sent.

Victim: **You must use Outlook on your CDN machine to open the email**.

* Open the email and, if you think the email is convincing enough, unzip it and execute the attachment.
* Provide screenshots of the open email and any security warnings.
* What happened after you ran the attachment, if anything was noticeable?

Blackhat:

* Provide a screenshot immediately after the session is established and you start interacting with your session.
* Display the hostname and operating system version of the victim’s machine. Provide a screenshot.

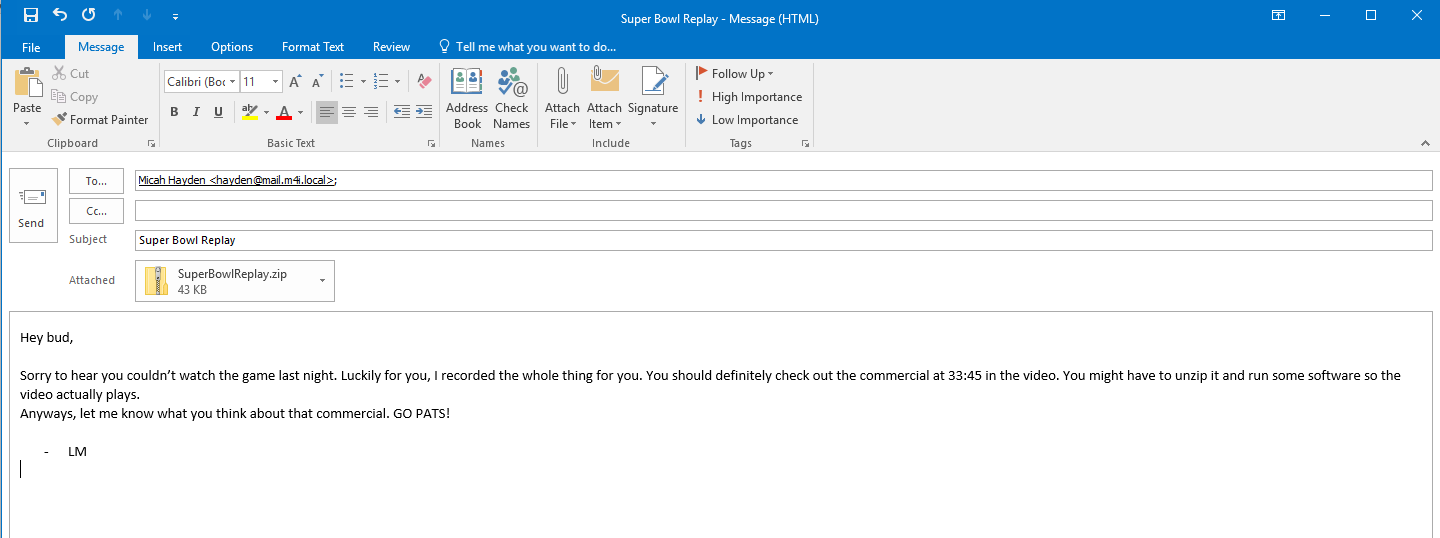
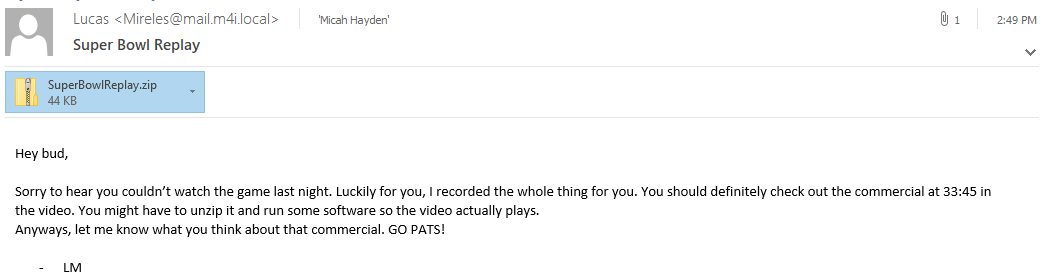
**Remember to turn off the proxy server in IE and remove Burp’s CA certificate!**

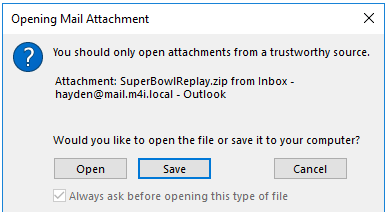
* Follow the instructions at the end of the document *Installing Burp CA certificate.docx*.

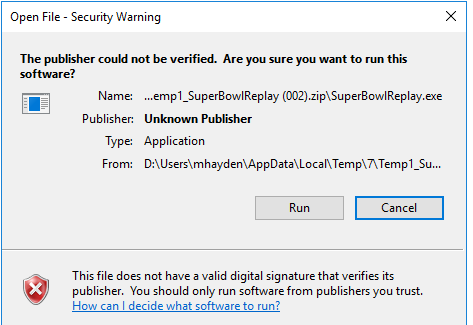
**Blackhat setup:** The blackhat used all options/commands shown in the slides.

* Options selected: Social-Engineering Attacks (1) 🡪 Create a Payload and Listener (4) 🡪 Payload – Windows Shell Reverse TCP (1)
* LHost – 10.204.0.43
* Port for the reverse listener – 88

**Email:**

* The following email was crafted on the Blackhat’s computer:
* The victim received the email as it appears below:
* The victim downloaded and ran the attachment, despite the following attachment download warning and run warning:

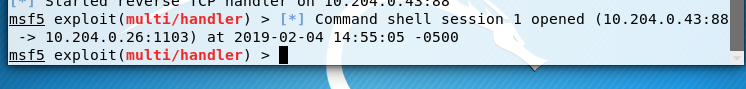




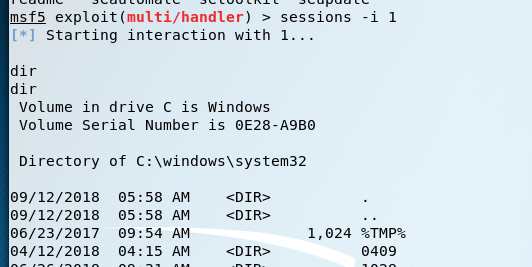
* After running the attachment, nothing occurred on the victim’s machine.

**Blackhat Computer:**

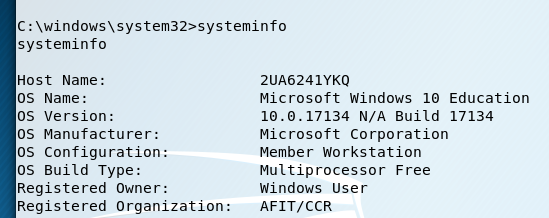
* In the Social Engineering Toolkit shell, the following was displayed to the console immediately after the victim ran the attachment:



* The Blackhat ran the following command to open/interact with the command shell session opened on the victim’s device:



* The Blackhat executed the command “systeminfo” which displayed the hostname and Operating System, outlined in red below:



**General Observations**

How long did it take you to complete this lab?

* This lab took around 8 hours to complete.

Was it an appropriate length lab?

* It was an appropriate length. However, there was a lot of tedious screenshots because of unclear requirements, detailed in the following question.

What corrections and or improvements do you suggest for this lab? Please be very specific, and if you add new material, provide the exact wording and instructions you would give to future students in the new lab handout. You may cross out and edit the text of the lab on previous pages to make minor corrections/suggestions.

* I would specify the differences in setup/actual lab questions. Question 1 contained a lot of information regarding what settings to use with Burp/IE, where the cookies are located, etc. I believe time would be better spent simply focusing on the session spoofing and the questions required, because the session spoofing will only work if the Burp setup and cookie information was done appropriately.
* Also with Questions 1-2: there were parts of the instructions that seemed to be questions, but they weren’t numbered with the other questions. I would clarify/separate **Instructions** and **Questions** to remove any confusion. If there are parts of the Instructions that require screenshots, indicate it as such, or add it to the list of questions.  
  **Example:** below is the text between question **a** and question **b** on Part 1.

***Now start Burp Proxy and configure IE to use the proxy for all traffic.***

* Download (<https://portswigger.net/burp/communitydownload>) and start Burp Proxy
  + Start the proxy listener on 127.0.0.1:8081 (Yes, use port 8081 because WebGoat uses 8080)
    - Proxy tab 🡪 Options tab 🡪 Proxy Listeners section 🡪 click on address 🡪 Edit 🡪 make changes 🡪 OK 🡪 Click the Running box next to the interface
  + Turn intercept off.
* Open IE and configure a proxy connection
  + Settings icon 🡪 Internet Options 🡪 Connections 🡪 LAN Settings
  + Check the box next to **Use a proxy server for your LAN** and enter 127.0.0.1 (assuming you’re running the proxy on the same machine as IE) as the address and port number 8081
* You need to install Burp Suite’s CA certificate. If you do not, IE complains about the Burp certificate not being trusted and blocks your connection.
  + Follow the instructions at https://support.portswigger.net/customer/portal/articles/1783075-installing-burp-s-ca-certificate-in-your-browser.
* Test the proxy connection by surfing to www.amazon.com. Since you are not intercepting anything yet, the proxy should be completely transparent and the Amazon page should be displayed.

The final bullet, “Test the proxy…” could appear as something which needs to be shown. We chose not to show that, because we assumed that none of the other questions could be accomplished if the Proxy was not working properly.